BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: 10/021,260

First Named Inventor: Carsten Driesner

Filed: March 14, 2005

Examiner: Nathan E. Price

Art Unit: 2194

Confirmation No.: 7197

Mail Stop: Appeal Brief
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

This Brief is submitted in support of an appeal from a final Office Action, mailed June 5, 2008, pursuant to a Notice of Appeal filed September 5, 2008.

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party in interest is Sun Microsystems, Inc., a corporation, with a place of business at 4150 Network Circle, Santa Clara, California 95054 .

RELATED APPEALS AND INTERFERENCES

Appellants are not aware of any related appeals or interferences.

STATUS OF CLAIMS

Claims 1-35 are pending. Claims 23-33 have been rejected under 35 U.S.C. 112 as failing to comply with the enablement requirement. Claims 23-33 have been rejected under 35 U.S.C. 101 as directed to non-statutory subject matter. Claims 1-14 and 20-35 have been rejected under 35 U.S.C §103(a) as allegedly being unpatentable over Coulouris, et al. (Coulouris, Distributed Systems Concepts and Design, 2d. ed., Addison-Wesley, 1994)("Coulouris") in view of Fidge (Fidge, "Logical Time in Distributed Computing Systems," Computer, Vol. 24, Issue 8, pp. 28-33, ISSN 0018-9162, August 1991)("Fidge"). Claims 15-19 have been rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Coulouris in view of Fidge and further in view of Liedtke (Liedtke, "Improving IPC by Kernal Design," ACM Symposium on Operating System Principles, Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, ACM Press, pp. 175-188, 1994)("Liedtke").

Claims 1-35 are the subject of this appeal.

STATUS OF AMENDMENTS

There are no presently pending amendments.

SUMMARY OF CLAIMED SUBJECT MATTER

Claim 1 reads as follows:

1. A method in a data processing system for synchronizing calls at a client in a server and client system, comprising the steps of:

receiving from the server a plurality of service calls generated by a plurality of threads executed at the server[201 and 204];

receiving a synchronization call from the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change[204]; and

placing at least one of said service calls associated with said synchronization call into a wait position, when said number of service calls indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ[205].

As apparent from the claim language, claim 1 recites receiving from the server a plurality of service calls generated by a plurality of threads executed at the server. See, e.g., Specification at ¶¶ 14, 22, 26-27, 30, 34, 38, 40, 42, and 44-48. In an illustrative example, the service calls may originate from threads of processes for a distributed application. See, e.g., Specification at 2 and 14. The distributed application may have modules with processes at both the client and the server, and cooperation may be necessary between the modules to run the distributed application. See, e.g., Specification at 2,14, and 20.

Next, a synchronization call is received that is a separate and different type of call from the service calls. See, e.g., Specification at 21-22. This differs from previous approaches that put a unique identifier in every call in order to specify the sequence of execution of calls and lead to a large communication overhead due to the inclusion of an identifier in every call. See, e.g., Specification at 21-22.

Continuing with the example, the separate synchronization call is transmitted to the client in order to allow the client to synchronize a service call execution when one of the threads executing at the server has changed. See, e.g., Specification at 27. The synchronization call may indicate that a thread has changed and indicate the number of service calls that have been generated by the server threads prior to the change. Specifically, claim 1 recites receiving a synchronization call from the server, the synchronization call being a separate and different type of call from the service calls and indicating that one of the plurality of threads executed at the server has changed and indicating a number of service calls

generated by the plurality of threads at the server prior to the thread change. See, e.g., Specification at ¶¶ 22, 26-27, 30, 34, 38-39, 40, 42, and 44-48.

The client may synchronize service call execution by knowing that a service call has changed and the number of service calls generated at the server prior to the change, and may place at least one of the service calls into a wait position to synchronize. See, e.g., Specification at ¶¶ 27. Claim 1 recites placing at least one of said service calls associated with said synchronization call into a wait position, when said number of service calls indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ. See, e.g., Specification at ¶¶ 16, 22-23, 25-28, 31-32, 35-36, 42-43, and 44-50.

12. A method in a data processing system for synchronizing calls at a server in a server and client system, comprising the steps of:

transmitting a plurality of service calls generated by a plurality of threads at the server to a client [201];

generating a synchronization call when a thread of said plurality of threads executed at the server changes, said synchronization call being a separate and different type of call from the service calls and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change[202]; and

transmitting said synchronization call to the client to allow the client to synchronize a service call execution[203].

As apparent from the claim language, claim 12 recites transmitting a plurality of service calls generated by a plurality of threads at the server to a client. See, e.g., Specification at ¶¶ 14, 22, 26-27, 30, 34, 38, 40, 42, and 44-48. In an illustrative example, the service calls may originate from threads of processes for a distributed application. See, e.g., Specification at 2 and 14. The distributed application may have modules with processes at both the client and the server, and cooperation may be necessary between the modules to run the distributed application. See, e.g., Specification at 2,14, and 20.

Next, a synchronization call is transmitted that is a separate and different type of call from the service calls. See, e.g., Specification at 21-22. This differs from previous

approaches that put a unique identifier in every call in order to specify the sequence of execution of calls and lead to a large communication overhead due to the inclusion of an identifier in every call. See, e.g., Specification at 21-22.

Continuing with the example, the separate synchronization call is transmitted to the client in order to allow the client to synchronize a service call execution when one of the threads executing at the server has changed. See, e.g., Specification at 27. The synchronization call may indicate that a thread has changed and indicate the number of service calls that have been generated by the server threads prior to the change. Specifically, claim 12 recites generating a synchronization call when a thread of said plurality of threads executed at the server changes, said synchronization call being a separate and different type of call from the service calls and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change. See, e.g., Specification at ¶¶ 22, 26-27, 30, 34, 38-39, 40, 42, and 44-48.

The client may synchronize service call execution by knowing that a service call has changed and the number of service calls generated at the server prior to the change, and may place at least one of the service calls into a wait position to synchronize. See, e.g., Specification at ¶¶ 27. Claim 12 recites transmitting said synchronization call to the client to allow the client to synchronize a service call execution. See, e.g., Specification at ¶¶ 16, 22-23, 25-28, 31-32, 35-36, 42-43, and 44-50.

22. A method in a data processing system for synchronizing calls in a client and server system, the method comprising the steps of:

transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client [201];

generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change[202];

transmitting said synchronization call to the client to allow the client to synchronize a service call execution[203];

receiving said synchronization call at the client[204]; and

placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ[205].

As apparent from the claim language, claim 22 recites transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client. See, e.g., Specification at ¶¶ 14, 22, 26-27, 30, 34, 38, 40, 42, and 44-48. In an illustrative example, the service calls may originate from threads of processes for a distributed application. See, e.g., Specification at 2 and 14. The distributed application may have modules with processes at both the client and the server, and cooperation may be necessary between the modules to run the distributed application. See, e.g., Specification at 2,14, and 20.

Next, a synchronization call is transmitted that is a separate and different type of call from the service calls. See, e.g., Specification at 21-22. This differs from previous approaches that put a unique identifier in every call in order to specify the sequence of execution of calls and lead to a large communication overhead due to the inclusion of an identifier in every call. See, e.g., Specification at 21-22.

Continuing with the example, the separate synchronization call is transmitted to the client and received at the client in order to allow the client to synchronize a service call execution when one of the threads executing at the server has changed. See, e.g., Specification at 27. The synchronization call may indicate that a thread has changed and indicate the number of service calls that have been generated by the server threads prior to the change. Specifically, claim 22 recites generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change. See, e.g., Specification at ¶¶ 22, 26-27, 30, 34, 38-39, 40, 42, and 44-48.

The client may synchronize service call execution by knowing that a service call has changed and the number of service calls generated at the server prior to the change, and may

place at least one of the service calls into a wait position to synchronize. See, e.g., Specification at ¶¶ 27. Claim 22 recites transmitting said synchronization call to the client to allow the client to synchronize a service call execution; receiving said synchronization call at the client; and placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ. See, e.g., Specification at ¶¶ 16, 22-23, 25-28, 31-32, 35-36, 42-43, and 44-50.

23. A computer readable storage medium containing instructions that cause a data processing system to perform a method of synchronizing calls in a client and a server system, the method comprising the steps of:

transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client [201];

generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change[202];

transmitting said synchronization call to the client to allow the client to synchronize a service call execution[203];

receiving said synchronization call at the client[204]; and

placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ[205].

As apparent from the claim language, claim 23 recites transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client. See, e.g., Specification at ¶¶ 14, 22, 26-27, 30, 34, 38, 40, 42, and 44-48. In an illustrative example, the service calls may originate from threads of processes for a distributed application. See, e.g., Specification at 2 and 14. The distributed application may have modules with processes at both the client and the server, and cooperation may be necessary between the modules to run the distributed application. See, e.g., Specification at 2,14, and 20.

Next, a synchronization call is transmitted that is a separate and different type of call from the service calls. See, e.g., Specification at 21-22. This differs from previous approaches that put a unique identifier in every call in order to specify the sequence of execution of calls and lead to a large communication overhead due to the inclusion of an identifier in every call. See, e.g., Specification at 21-22.

Continuing with the example, the separate synchronization call is transmitted to the client and received at the client in order to allow the client to synchronize a service call execution when one of the threads executing at the server has changed. See, e.g., Specification at 27. The synchronization call may indicate that a thread has changed and indicate the number of service calls that have been generated by the server threads prior to the change. Specifically, claim 23 recites generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change. See, e.g., Specification at ¶¶ 22, 26-27, 30, 34, 38-39, 40, 42, and 44-48.

The client may synchronize service call execution by knowing that a service call has changed and the number of service calls generated at the server prior to the change, and may place at least one of the service calls into a wait position to synchronize. See, e.g., Specification at ¶¶ 27. Claim 23 recites transmitting said synchronization call to the client to allow the client to synchronize a service call execution; receiving said synchronization call at the client; and placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ. See, e.g., Specification at ¶¶ 16, 22-23, 25-28, 31-32, 35-36, 42-43, and 44-50.

34. A data processing system for synchronizing calls in a client and server system, the data processing system comprising:

a client computer [20] comprising:

a memory [22] including a client program [23] that receives a plurality of service calls [32] generated by a plurality of threads executed at the server [10], that receives a synchronization call [31] from the server, said synchronization call being a separate and different type of call from the service calls [32] and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls [28] generated by said plurality of threads at the server prior to the thread change, and that places at least one of said service calls associated with said synchronization call into a wait position [29], if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ; and

a first processor [21] that runs said client program;

a server computer comprising:

a memory [22] including a server program [15] that transmits the plurality of service calls generated by the plurality of threads at the server to the client, that generates the synchronization call when a thread of said plurality of threads executed at the server changes, said synchronization call indicating a number of service calls generated by said plurality of threads at the server prior to the thread change, and that transmits said synchronization call to the client to allow the client to synchronize a service call execution; and

a second processor [11] that runs said server program; and

a network [30] connecting said client computer and said server computer.

As apparent from the claim language, claim 34 recites a client computer comprising: a memory including a client program that receives a plurality of service calls generated by a plurality of threads executed at the server. . . a first processor that runs said client program. See, e.g., Specification at ¶¶ 14, 22, 26-27, 30, 34, 38, 40, 42, and 44-48. In an illustrative example, the service calls may originate from threads of processes for a distributed application. See, e.g., Specification at 2 and 14. The distributed application may have modules with processes at both the client and the server, and cooperation may be necessary between the modules to run the distributed application. See, e.g., Specification at 2,14, and 20.

Next, a synchronization call is transmitted that is a separate and different type of call from the service calls. See, e.g., Specification at 21-22. This differs from previous

approaches that put a unique identifier in every call in order to specify the sequence of execution of calls and lead to a large communication overhead due to the inclusion of an identifier in every call. See, e.g., Specification at 21-22.

Continuing with the example, the separate synchronization call is transmitted to the client and received at the client in order to allow the client to synchronize a service call execution when one of the threads executing at the server has changed. See, e.g., Specification at 27. The synchronization call may indicate that a thread has changed and indicate the number of service calls that have been generated by the server threads prior to the change. Specifically, claim 34 recites receives a synchronization call from the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change, and that places at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ. See, e.g., Specification at ¶¶ 22, 26-27, 30, 34, 38-39, 40, 42, and 44-48.

The client may synchronize service call execution by knowing that a service call has changed and the number of service calls generated at the server prior to the change, and may place at least one of the service calls into a wait position to synchronize. See, e.g., Specification at ¶¶ 27. Claim 34 recites a memory including a server program that transmits the plurality of service calls generated by the plurality of threads at the server to the client, that generates the synchronization call when a thread of said plurality of threads executed at the server changes, said synchronization call indicating a number of service calls generated by said plurality of threads at the server prior to the thread change, and that transmits said synchronization call to the client to allow the client to synchronize a service call execution; and a second processor that runs said server program; and a network connecting said client computer and said server computer. See, e.g., Specification at ¶¶ 16, 22-23, 25-28, 31-32, 35-36, 42-43, and 44-50.

35. An apparatus for synchronizing calls in a client and server system, the apparatus comprising:

means for transmitting a plurality of service calls [15, 30, 32] generated by a plurality of threads executed at the server[15, 20] to the client [10];

means for generating a synchronization call at the server [15, 20, 22], said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change;

means for transmitting said synchronization call to the client to allow the client to synchronize a service call execution [30];

means for receiving said synchronization call at the client[20, 22]; and

means for placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ [15, 22].

As apparent from the claim language, claim 35 recites means for transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client. See, e.g., Specification at ¶¶ 14, 22, 26-27, 30, 34, 38, 40, 42, and 44-48. In an illustrative example, the service calls may originate from threads of processes for a distributed application. See, e.g., Specification at 2 and 14. The distributed application may have modules with processes at both the client and the server, and cooperation may be necessary between the modules to run the distributed application. See, e.g., Specification at 2,14, and 20.

Next, a synchronization call is transmitted that is a separate and different type of call from the service calls. See, e.g., Specification at 21-22. This differs from previous approaches that put a unique identifier in every call in order to specify the sequence of execution of calls and lead to a large communication overhead due to the inclusion of an identifier in every call. See, e.g., Specification at 21-22.

Continuing with the example, the separate synchronization call is transmitted to the client and received at the client in order to allow the client to synchronize a service call execution when one of the threads executing at the server has changed. See, e.g., Specification at 27. The synchronization call may indicate that a thread has changed and indicate the number of service calls that have been generated by the server threads prior to the change. Specifically, claim 35 recites means for generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change; means for transmitting said synchronization call to the client to allow the client to synchronize a service call execution; and means for receiving said synchronization call at the client. See, e.g., Specification at ¶¶ 22, 26-27, 30, 34, 38-39, 40, 42, and 44-48.

The client may synchronize service call execution by knowing that a service call has changed and the number of service calls generated at the server prior to the change, and may place at least one of the service calls into a wait position to synchronize. See, e.g., Specification at ¶¶ 27. Claim 35 means for placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ. See, e.g., Specification at ¶¶ 16, 22-23, 25-28, 31-32, 35-36, 42-43, and 44-50.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 23-33 have been rejected under 35 U.S.C. 112 as failing to comply with the enablement requirement. Claims 23-33 have been rejected under 35 U.S.C. 101 as directed to non-statutory subject matter. Claims 1-14 and 20-35 have been rejected under 35 U.S.C §103(a) as allegedly being unpatentable over Coulouris in view of Fidge. Claims 15-19 have been rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Coulouris in view of Fidge and further in view of Liedtke.

Review of all grounds of rejection is sought on this appeal.

ARGUMENT

1. The present specification enables one skilled in the art to make and use the claimed invention.

The enablement requirement refers to the requirement of 35 USC 112, first paragraph that the specification describe how to make and how to use the invention. The purpose of this requirement is to ensure that the invention is communicated to the interested public in a meaningful way. The test for determining whether or not the specification meets this requirement is whether the information contained in the specification is sufficient to inform those skilled in the relevant art how to make and use the claimed invention.

In the present case, claims 23 - 33 recite a computer readable storage medium containing instructions that cause a data processing system to perform a particular method of synchronizing calls in a client and a server system, the method being made up of specifically recited activities. That is, the claims recite a particular article of manufacture (i.e., a computer readable storage medium) having certain instructions contained therein. In support of these claims, the specification describes particular examples of computer readable storage mediums, for example hard disks, floppy disks, CD-ROM, and other forms of ROM and RAM. See Specification at p. 21, para. 3. Such storage mediums are well known in the computer-related arts and indeed even among lay persons. The manner of storing instructions in such devices (e.g., storing software on a hard disk or in memory, or storing a computer file onto a CD-ROM or other storage medium) is also well known among those who practice in these arts and, indeed, among lay persons. Many, if not most, adults and even children in the United States are quite familiar with the notion of storing a file on such a medium. Further, the precise steps that make up the method which is instantiated in the instructions are described in detail in the specification, for example, see Specification at p.26-28. Coding of such processes is an activity well within the ordinary skill of those who practice in the relevant art, hence, the specification by teaching examples of particular storage media which can be used and describing in detail the kinds of processes that would have to be coded as instructions for

storage on these media has completely and fully described how to make and use the invention recited in these claims. Consequently, the rejections should be reversed.

2. **The claimed invention is directed to statutory subject matter.**

Claims 23-33 are directed to statutory subject matter with the recitation of "computer readable storage medium containing instructions" in the preamble of the claims. The specification at page 21, para. 3, provides examples of computer readable medium storage devices (e.g. hard disks, floppy disks, CD-ROM). By reciting "storage medium", the claimed invention is directed toward to tangible mediums as opposed to carrier waves that are transitory in nature. Therefore, the claimed invention is directed to statutory subject matter.

3. **The present claims are patentable over the combination of Coulouris in view of Fidge, which fails to disclose or suggest methods and systems with a synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change, as presently claimed.**

Coulouris fails to disclose or suggest a synchronization call that is a separate and different type of call from service calls, as recited in the independent claims. As acknowledged by the Examiner in the Office Actions dated June 15, 2007 and June 5, 2008, Coulouris' alleged synchronization calls and service calls are the same. In other words, Coulouris' alleged synchronization calls are the same type of calls as its service calls, and therefore, the Coulouris synchronization calls are not separate and different type of call from a service call.

Further, none of the cited passages suggest a synchronization call that is separate and different type from Coulouris' other messages. The passages cited by the Examiner relate to two types of subject matter: 1) timestamps and 2) remote procedure call ordering.

On the subject of timestamps, Coulouris describes that every message carries a timestamp to ensure the messages are placed in proper order upon receipt (Coulouris 326, 342, and 396-397). To the extent that the Examiner considers the existence of a

timestamp in a message to make the message act as a service call and a synchronization call, only one type of message (e.g. a timestamped message) is taught in Coulouris. Thus, Coulouris does not suggest a synchronization call that is separate and different type of call from a service call.

Nowhere in the cited Coulouris' discussion of remote procedure calls (RPCs) does Coulouris disclose or suggest a synchronization call, which is a separate and different type from service calls and identifies 1) whether a thread has changed or 2) the number of calls generated by threads prior to the thread change. Instead, Coulouris discloses that a "remote procedure call" (RPC) can be an asynchronous procedure call that does not require a reply from the server (Coulouris p.150- 151). Coulouris does not specify the content of the remote procedure calls, much less a synchronization call that identifies whether a thread has changed or the number of calls generated by threads prior to thread change. Thus, Coulouris does not disclose or suggest methods and systems with a synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change.

Thus, Coulouris fails to disclose or suggest Applicants' claimed synchronization call.

Fidge also fails to disclose or suggest Applicants' claimed synchronization call. The Examiner argues that Fidge at page 30, Rules B and F, discloses or suggest Applicants' claimed synchronization call. Instead, Fidge describes a method for ordering messages by using counter values attached to every message (Fidge 29). Each message includes a numerical counter value associated with the process instance that sends the message. Id. Nowhere does Fidge disclose or suggest a synchronization call, which is separate from service calls. Instead, Fidge merely teaches one type of message that includes a counter value associated with the sending processing instance.

Hence, even if the teachings of Coulouris are combined with Fidge, the present invention would still include features not provided by the references. For at least these reasons, all of the present claims are patentable over Coulouris in view of Fidge.

4. **The present claims are patentable over the combination of Coulouris in view of Fidge in further view of Liedtke, which fails to disclose or suggest methods and systems with a synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change, as presently claimed.**

Independent claim 12 is patentable over Coulouris in view of Fidge as discussed above. Liedtke fails to disclose or suggest Applicants' claimed synchronization call that is separate from service calls. Liedtke is directed toward remote procedure calls. Therefore, Coulouris in view of Fidge and further in view of Liedtke still fails to disclose or suggest claim 12.

Claims 15-19 depend directly or indirectly from claim 12 and are therefore allowable for at least the same reasons that claim 12 is allowable.

Hence, even if the teachings of Coulouris are combined with Fidge in further view of Liedtke, the present invention would still include features not provided by the references. For at least these reasons, all of the present claims are patentable over the cited references.

CONCLUSION

For the foregoing reasons, reversal of the Examiner's rejections as set forth in the final Office Action with respect to claims 1-35 is respectfully requested. If there are any additional fees due in connection with this communication, please charge Deposit Account No. 19-3140.

Respectfully submitted,

SONNENSCHEIN NATH & ROSENTHAL LLP

Dated: November 5, 2008

/Tarek N. Fahmi/
Tarek N. Fahmi
Reg. No. 41,402

PO Box 061080
Wacker Drive Station
Sears Tower
Chicago, IL 60606-1080
(650) 798-0320

CLAIMS APPENDIX

1. A method in a data processing system for synchronizing calls at a client in a server and client system, comprising the steps of:

receiving from the server a plurality of service calls generated by a plurality of threads executed at the server;

receiving a synchronization call from the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change; and

placing at least one of said service calls associated with said synchronization call into a wait position, when said number of service calls indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ.

2. The method according to claim 1, wherein said service calls are associated with said synchronization call by one of including respective identifiers into said at least one of said synchronization call and said service calls, and indicating one of a specific reception sequence and order of service of said service calls and said at least one synchronization call at the client.

3. The method of synchronizing calls according to claim 1, wherein said receiving steps include receiving a first call sequence of a plurality of call sequences from the server, said first call sequence including a first synchronization call and at least one service call from a first thread, said first synchronization call including a first server call counter value indicating a first number of service calls executed at the server prior to the first synchronization call;

said method further comprising the step of:

comparing said first server call counter value with a client call counter value, said client call counter value indicating a second number of service calls executed at the client prior to receiving said first synchronization call; and

one of:

executing said first number of service calls of said first call sequence and counting said executed first number of service calls using a client call counter value, if said client call counter value and said first server call counter value coincide; and

placing said first call sequence into a wait position, if said client call counter value and said first server current call counter value differ.

4. The method according to claim 1, wherein said service calls are generated asynchronously.

5. The method according to claim 3, further comprising the steps of:

determining whether a second call sequence in a wait position is available, said second call sequence including a plurality of service calls from a second thread executed at the server and a second synchronization call including a second server call counter value indicating a third number of service calls executed at the server prior to said second synchronization call;

wherein if said second call sequence in a wait position is not available, waiting to receive further service calls and synchronization calls; and

wherein if said second call sequence is available, determining that said second server call counter value coincides with said client call counter value, and executing said third number of service calls of said second call sequence and incrementing said client counter value for each executed third number of service calls.

6. The method of according to claim 5, further comprising the step of:

waiting for a third call sequence to be received from the server unit, the third call sequence including a third synchronization call including a third server call counter value coinciding with said client call counter value.

7. The method according to claim 3, wherein said call sequences are received as groups included into packets from the server, each group being generated upon one of a timer signal at the server, a synchronous call at the server, and a synchronization call at the server.

8. The method according to claim 2, wherein said synchronization call and said service calls are received in an arbitrary order.

9. The method according to claim 1, wherein said service calls from said plurality of threads at the server are executed in corresponding threads at the client.

10. The method according to claim 3, wherein said first server call counter value indicates a total number of service calls at the server executed prior to a current service call and requires communication with the client; and

wherein said client call counter value indicates a total number of service calls executed at the client and involves communication with the server.

11. The method according to claim 1, wherein each of said service calls from the server includes at least one of:

obtaining instructions to display information on a display of the client;

rendering instructions;

storing instructions to store information at the client; and

information on processing results from the server.

12. A method in a data processing system for synchronizing calls at a server in a server and client system, comprising the steps of:

transmitting a plurality of service calls generated by a plurality of threads at the server to a client;

generating a synchronization call when a thread of said plurality of threads executed at the server changes, said synchronization call being a separate and different type of call from the service calls and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change; and

transmitting said synchronization call to the client to allow the client to synchronize a service call execution.

13. The method according to claim 12, wherein said service calls are associated with said synchronization call by one of including respective identifiers into said at least one of said synchronization call and said service calls, and indicating one of a specific reception sequence and order of service of said service calls and said at least one synchronization call at the client.

14. The method according to claim 12, wherein said service calls are generated asynchronously.

15. The method according to claim 12, further comprising the steps of:

generating a current service call by a first thread executed at the server;

determining a first thread identifier of a first thread and comparing said first thread ID with a second thread identifier of a second thread which issued a service call preceding said current service call;

wherein, if said first thread identifier and said second thread identifier differ, generating a first synchronization call including a server call counter value indicating a number of service calls executed at the server prior to said current service call and transmitting said first synchronization call to the client, for enabling the client to synchronize an execution of a plurality of service calls from at least said first thread and said second thread; and

counting said current service call using said server call counter value if said first thread identifier and said second thread identifier do not differ.

16. The method according to claim 15, wherein a plurality of service calls from said first thread and said synchronization call comprise a call sequence; and

wherein said call sequences are received as groups included into packets from the server, each group being generated upon one of a timer signal at the server, a synchronous call at the server, and a synchronization call at the server

17. The method according to claim 15, wherein said synchronization call includes said second thread identifier of said second thread, and said number of service calls include a thread identifier of each thread generating said service call; and

wherein said synchronization call and said number of service calls are transmitted to the client in an arbitrary order.

18. The method according to claim 15, wherein said service calls from said plurality of threads at the server are executed in corresponding threads at the client.

19. The method according to claim 15, wherein said server call counter value indicates a total number of service calls requiring communication with the client executed at the server, prior to the current service call.

20. The method according to claim 12, wherein each service call from the server includes at least one of:

obtaining instructions to display information on a display of the client;

rendering instructions;

storing instructions to store information at the client; and

information on processing results from the server.

21. The method according to claim 12, wherein a synchronization call is further generated upon an occurrence of one of the group consisting of:

a timer signal;

a predetermined number of service calls; and

a synchronous call.

22. A method in a data processing system for synchronizing calls in a client and server system, the method comprising the steps of:

transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client;

generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change;

transmitting said synchronization call to the client to allow the client to synchronize a service call execution;

receiving said synchronization call at the client; and

placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ.

23. A computer readable storage medium containing instructions that cause a data processing system to perform a method of synchronizing calls in a client and a server system, the method comprising the steps of:

transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client;

generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change;

transmitting said synchronization call to the client to allow the client to synchronize a service call execution;

receiving said synchronization call at the client; and

placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ.

24. The computer readable storage medium according to claim 23, wherein said service calls are associated with said synchronization call by one of including respective identifiers into said at least one of said synchronization call and said service calls, and indicating one of a specific reception sequence and order of service of said service calls and said at least one synchronization call at the client.

25. The computer readable storage medium according to claim 24, wherein said receiving step includes receiving a first call sequence of a plurality of call sequences from the server, said first call sequence including a first synchronization call and at least one service call from a first thread, said first synchronization call including a first server call counter value indicating a first number of service calls executed at the server prior to the first synchronization call;

said method further comprising the step of:

comparing said first server call counter value with a client call counter value, said client call counter value indicating a second number of service calls executed at the client prior to receiving said first synchronization call; and

one of:

executing said first number of service calls of said first call sequence and counting said executed first number of service calls using a client call counter value, if said client call counter value and said first server call counter value coincide; and

placing said first call sequence into a wait position, if said client call counter value and said first server current call counter value differ.

26. The computer readable storage medium according to claim 24, wherein said service calls are generated asynchronously.

27. The computer readable storage medium according to claim 26, further comprising the steps of:

determining whether a second call sequence in a wait position is available, said second call sequence including a plurality of service calls from a second thread executed at the server

and a second synchronization call including a second server call counter value indicating a third number of service calls executed at the server prior to said second synchronization call;

wherein if said second call sequence in a wait position is not available, waiting to receive further service calls and synchronization calls; and

wherein if said second call sequence is available, determining that said second server call counter value coincides with said client call counter value, and executing said third number of service calls of said second call sequence and incrementing said client counter value for each executed third number of service calls.

28. The computer readable storage medium according to claim 27, further comprising the step of:

waiting for a third call sequence to be received from the server unit, the third call sequence including a third synchronization call including a third server call counter value coinciding with said client call counter value.

29. The computer readable storage medium according to claim 26, wherein said call sequences are received as groups included into packets from the server, each group being generated upon one of a timer signal at the server, a synchronous call at the server, and a synchronization call at the server.

30. The computer readable storage medium according to claim 26, wherein said synchronization call and said service calls are received in an arbitrary order.

31. The computer readable storage medium according to claim 24, wherein said service calls from said plurality of threads at the server are executed in corresponding threads at the client.

32. The computer readable storage medium according to claim 26, wherein said first server call counter value indicates a total number of service calls at the server executed prior to a current service call and requires communication with the client; and

wherein said client call counter value indicates a total number of service calls executed at the client and involves communication with the server.

33. The computer readable storage medium according to claim 24, wherein each of said service calls from the server includes at least one of:

obtaining instructions to display information on a display of the client;

rendering instructions;

storing instructions to store information at the client; and

information on processing results from the server.

34. A data processing system for synchronizing calls in a client and server system, the data processing system comprising:

a client computer comprising:

a memory including a client program that receives a plurality of service calls generated by a plurality of threads executed at the server, that receives a synchronization call from the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change, and that places at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ; and

a first processor that runs said client program;

a server computer comprising:

a memory including a server program that transmits the a plurality of service calls generated by the a plurality of threads at the server to the client, that generates the a synchronization call when a thread of said plurality of threads executed at the server changes, said synchronization call indicating a number of service calls generated by said plurality of threads at the server prior to the thread change, and that transmits said synchronization call to the client to allow the client to synchronize a service call execution; and

a second processor that runs said server program; and

a network connecting said client computer and said server computer.

35. An apparatus for synchronizing calls in a client and server system, the apparatus comprising:

means for transmitting a plurality of service calls generated by a plurality of threads executed at the server to the client;

means for generating a synchronization call at the server, said synchronization call being a separate and different type of call from the service calls and indicating that one of said

plurality of threads executed at the server has changed and indicating a number of service calls generated by said plurality of threads at the server prior to the thread change;

means for transmitting said synchronization call to the client to allow the client to synchronize a service call execution;

means for receiving said synchronization call at the client; and

means for placing at least one of said service calls associated with said synchronization call into a wait position, if said number indicated in said synchronization call and said number of service calls executed at the client prior to receiving said synchronization call differ.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.